

Re: [PATCH] [2.4] forcedeth network driver

Source: <http://linux.derkeiler.com/Mailing-Lists/Kernel/2004-01/6287.html>

From: Carl-Daniel Hailfinger (*c-d.hailfinger.kernel.2004_at_gmx.net*)

Date: 01/24/04

Date: Sat, 24 Jan 2004 22:55:11 +0100

To: Jeff Garzik <jgarzik@pobox.com>

Jeff Garzik wrote:

> *Manfred Spraul wrote:*

>

>> *Jeff wrote:*

>>

>>> ** Interrupt handler is SCARY. You can potentially take and release*

>>> *the spinlock –many times– during a single interrupt.*

>>>

>> *I think that can happen only in theory: A new packet is completed*

>> *while the driver processes rx packets. In all normal cases there should*

>> *be one spinlock operation per tx irq, and 0 per rx irq.*

>> *And error handling IMHO doesn't count: it should be rare.*

>> *Or do I overlook a common case?*

>

>

> *Any amount of load at all will lead to multiple iterations of the*

> *master loop in the interrupt handler.*

```
+static int max_interrupt_work = 5;
[...]
```

```
+ for (i=0; ; i++) {
+   if (i > max_interrupt_work) {
+     spin_lock(&np->lock);
+     /* disable interrupts on the nic */
+     writel(0, base + NvRegIrqMask);
+     pci_push(base);
+
+     if (!np->in_shutdown)
+       mod_timer(&np->nic_poll, jiffies + POLL_WAIT);
+     printk(KERN_DEBUG "%s: too many iterations (%d) in
nic_irq.\n", dev->name, i);
+     spin_unlock(&np->lock);
+     break;
+   }
```

So actually it should not happen that often since we exit the master loop after 5 iterations.

Linux-Kernel: Re: [PATCH] [2.4] forcedeth network driver

```
>>>> +#define NV_MIIPHY_DELAY 10
>>>> +#define NV_MIIPHY_DELAYMAX 10000
>>>
>>>
>>>
>>> Style: it's fairly silly to mix enums and constants.
>>>
>>>
>> Right now: enum for the nic registers, #define for the rest. If you
>> don't like it I can change it.
>
>
> enums are definitely preferred... communicates more type/symbol
> information to the compiler and more symbol info to the debugger.
```

The current order has the benefit that the values which might be written to a register (most of which are constants) are located next to the enum for the register. Anyways, I have no strong opinion about it.

```
>>>> /* General driver defaults */
>>>> +#define NV_WATCHDOG_TIMEO (2*HZ)
>>>>
>>>>
>>>>
>>> this seems too short, and might trigger on normal events?
>>>
>>>
>> I think I copied it from another driver – which value would you
>> recommend?
>
>
> 5 seconds is the norm, but it also depends on whether your link
> interrupt is 100% reliable. If you don't have to synchronize the link
> watchdog timeout with other driver-private timers, the task is easier.
```

Hardware availability problem. We always have to assume something works until some user/tester complains.

```
>>> skb_reserve() seems to be missing
>>>
>>>
>>> Do you have specs that show that all nForce versions support unaligned
>>> buffers? skb_reserve is a performance feature, I don't want to add it
>>> yet. Testing that it works is on our TODO list.
>
>
> hmmmm, is nForce ever found on non-x86 boxes? I would think that
> skb_reserve might be –required– for some platforms.
```

Since nForce is an Athlon/Athlon64 chipset, I assume it is not found on any non-x86/x86-64 boxes. That might change, however, if NVidia licenses

their nic core to other vendors. So far, they have not given any evidence supporting that.

I have seen some hints that earlier nForce versions do NOT support unaligned buffers.

```
>>>> +/*
>>>> + * change_mtu: dev->change_mtu function
>>>> + * Called with dev_base_lock held for read.
>>>> + */
>>>> +static int change_mtu(struct net_device *dev, int new_mtu)
>>>> +{
>>>> + if (new_mtu > DEFAULT_MTU)
>>>> + return -EINVAL;
>>>> + dev->mtu = new_mtu;
>>>> + return 0;
>>>> +}
>>>
>>>
>>>
>>> bug #1: have you tested changing the MTU while the NIC is actually
>>> running?
>>>
>> What should the nic do? I'll continue to allocate 1.8 kB buffers
>> because I don't know how to reconfigure the nic hardware to reject
>> large packets.
```

I think the spec says something about it, but I'd have to get my hands on the hardware to test if it actually works.

> Fair enough. You may wish to (after testing!) increase DEFAULT_MTU by 4
> bytes, to support VLAN.

Already tried. If we increase the length of the packet the nic has to send beyond 1500 bytes, we simply get an TX error/timeout. So far, I have not seen any hints that the nic might do VLAN tagging internally.

Jeff, there is another bug in some hardware revisions: For every received packet it reports 1500 bytes length regardless of the real length. We take this len as correct and hope for the best. At best, it ruins our RX statistics. At worst, it might leak kernel memory to userspace since the unused remainder of the 1500 bytes is not initialized with anything, yet passed up with netif_rx(skb). Does the networking core provide any built-in function to address this problem?

```
+ prd = &np->rx_ring[i];
[...]
+ len = le16_to_cpu(prd->Length);
[...]
+ skb_put(skb, len);
[...]
```

Linux-Kernel: Re: [PATCH] [2.4] forcedeth network driver

```
+ netif_rx(skb);
```

The above code would be correct if the hardware actually gave us the right values. I know that at least some nForce2 systems suffer from this.

Carl-Daniel

—

To unsubscribe from this list: send the line "unsubscribe linux-kernel" in the body of a message to majordomo@vger.kernel.org

More majordomo info at <http://vger.kernel.org/majordomo-info.html>

Please read the FAQ at <http://www.tux.org/lkml/>